Flow-Through-Queue based Power Management for Gigabit Ethernet Controller[1]

Hwisung Jung
Dept. of Electrical Engineering
University of Southern California
Los Angeles, CA 90089
e-mail: hwijung@usc.edu

Andy Hwang
Enterprise Networking Group
Broadcom Corporation
Irvine, CA 92618
e-mail: ahwang@broadcom.com

Massoud Pedram
Dept. of Electrical Engineering
University of Southern California
Los Angeles, CA 90089
e-mail: pedram@usc.edu

Abstract - This paper presents a novel architectural mechanism and a power management structure for the design of an energy-efficient Gigabit Ethernet controller. Key characteristics of such a controller are low-latency and high-bandwidth required to meet the pressing demands of extremely high frame and control data, which in turn cause difficulties in managing power dissipation. We propose a flow-through-queue (FTQ) based power management method, which allows some of the tasks involved in processing the frame data to be offloaded. This in turn enables utilization of multiple clock rates and multiple voltages for different cores inside the Ethernet controller. A modeling approach based on semi-Markov decision process (SMDP) and queuing models is employed, which allow one to apply mathematical programming formulations for energy optimization under performance constraints. The proposed Gigabit Ethernet controller is designed with a 130nm CMOS technology that includes both high and low threshold voltages. Experimental results show that the proposed power optimization method can achieve system-wide energy savings under tighter performance constraints.

I. INTRODUCTION

A look at today's high-speed networking system trends reveals that as Internet link speeds continue to grow exponentially, a Gigabit Ethernet controller is becoming more complex to satisfy the high-functionality, high-performance demands of today's applications. For example, the Gigabit Ethernet controller must be able to support high frame-rate data processing and low-latency access to achieve full-duplex line rates for maximum-sized, e.g., 1518-byte, frame [1]. However, this trend also translates into high power densities, higher operating temperatures, and lower circuit reliability. Power consumption increases rapidly with increase in link speed [2]. Thus, designers of the Gigabit Ethernet controller must consider power dissipation as one of the primary issues.

Although power savings are commonly achieved through circuit-level optimization techniques, many opportunities exist at the system and architecture levels to reduce energy consumption. Furthermore, current CMOS technologies allow an increasing number of clock and voltage domains to be specified on the same chip, which allows dynamic voltage and frequency scaling (DVFS) and multiple supply and threshold voltage (V_{dd} and V_{th}) assignments to be utilized [7]. System designs utilizing multiple clocks and multiple voltage cores, where globally asynchronous and locally synchronous (GALS) communication architecture is deployed, face increasing difficulty in managing power consumption under tighter performance constraints [4][8].

As reported in [3]-[6], the problem of power modeling and optimization at high-levels of abstraction in GALS has received a lot of attention especially with respect to multiple voltage domains. In [3], the authors show that GALS processors with multiple clocks and a single voltage are not necessarily better in terms of power consumption compared to fully synchronous design due to the asynchronous communication overhead. It is also reported that the use of dynamic voltage scaling in multiple voltage cores improves power savings up to 20%. The work presented in [4] studies online DVFS scheme in the context of a multiple clock domain architecture by utilizing interface queues to guide the DVFS control. Voltage island-based power management is proposed in [5] to satisfy the required performance in multi-threshold CMOS technologies. In [6], the authors present an architecture for GALS systems, which allows dynamic load-balancing and adaptive inter-task voltage scaling based on the load in each of the processing units.

Although these techniques perform DVFS, little attention has been given to modeling a power-managed system with multiple V_{dd}/V_{th} choices. Indeed, a centralized DVFS architecture [3] that utilizes interface queues to transfer high-bandwidth data between multiple voltage domains tends to perform rather poorly under tight performance constraints. Finally, GALS [6] often results in overhead penalty in terms of timing due to the complexity of configurations.

In this paper, we propose a flow-through-queue (FTQ) based power management method by offloading some of the tasks involved in processing the frame data, which enables multiple clock rates and multiple voltage cores inside the Ethernet controller chip. Note that in the Gigabit Ethernet controller, the control data must be accessed with low-latency, while the frame data must be accessed with high bandwidth so as to maximize the transfer speed. These two competing requirements create a very challenging power minimization problem. FTQ, which directs the frame data processing between functional modules, improves hardware support for higher performance with respect to handling the incoming packets. We also present a systematic approach for constructing a stochastic power management model. The numerical optimization solution of this stochastic model is based on a semi-Markov decision process (SMDP). Note that SMDP model, which offers a robust theoretical framework, enables one to apply strong mathematical optimization techniques to derive optimal power management policies. To achieve further energy savings in multi-threshold CMOS technologies, mathematical programming problems are

[1]This work was supported in part by a grant (# 0541469) from the CCF division of the NSF.

formulated with multiple V_{dd}/V_{th} assignments under tight performance constraints.

The remainder of this paper is organized as follows: Section 2 provides a brief background of the Gigabit Ethernet controller while section 3 describes the details of proposed FTQ-based architecture. In section 4, we construct the FTQ-based system with SMDP and queuing models. Section 5 provides performance optimization methods. Experimental results and conclusion are given in section 6 and section 7.

II. BACKGROUND: ETHERNET CONTROLLER

The host system of a networking server uses the Ethernet controller to send and receive packets. Sending and receiving packets over the local interconnect, e.g., PCI-E bus [9], is handled by the Ethernet controller and the device driver in the host operating system. In general, the Ethernet controller typically has a direct memory access (DMA) engine to transfer data between the host system memory and the network interface memory. In addition, Ethernet controller includes a medium access control (MAC) unit to implement the link level protocol for the underlying network, and use a signal processing hardware to implement the physical (PHY) layer defined in the network. Figure 1 shows a simplified block diagram of the Gigabit Ethernet controller.

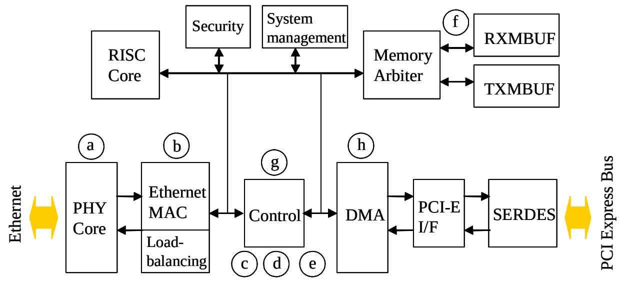


Figure 1. Block diagram of Gigabit Ethernet controller.

To understand the functionality of the Ethernet controller inside, the process of receiving a packet over the network is explained next (various steps are shown in Figure 1 in alphabetical orders). In step (a), the Ethernet controller receives a data stream from the selected physical layer interface. It performs address checking, CRC calculation, and CSMA/CD functions [2] in step (b). In step (c), the Ethernet controller calculates checksum and parses TCP/IP headers, while classifying frame based on a set of matching rules in step (d). In step (e), the Ethernet controller strips the VLAN (Virtual Local Area Network) tag, and then temporarily places packet data and header into the pre-allocated receive buffer (i.e., RXMBUF) in step (f). After that, the Ethernet controller completes buffer descriptors for the packet in step (g). Finally in step (h), the DMA transfer for packet data and descriptors to the host memory is accomplished via the PCI-E interface by notifying the device driver by means of an interrupt. Further details of the Gigabit Ethernet architecture and functionality are omitted to save space. Interested readers may refer to [10][11].

III. FTQ-BASED ARCHITECTURE

As described before, defragmenting the packets of various communication protocols in hardware remains an extremely complex task. Thus, the Ethernet controller needs more functional modules and specialized hardware units that efficiently transfer between the local interconnect and the network. Therefore, for a power-constrained system, it is necessary to capture parallelism and asynchrony among multiple functional modules operating at multiple clocks and voltages.

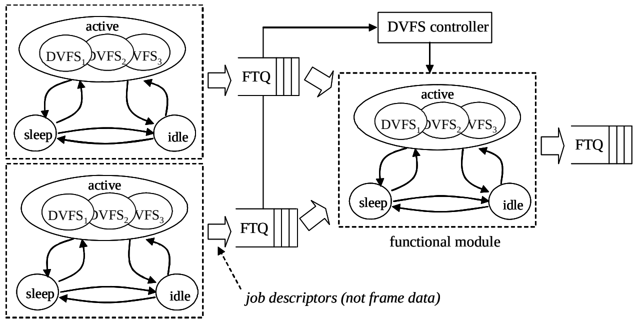


Figure 2. Concept of Flow-Through-Queue.

The FTQs provide a FIFO mechanism between the state machines describing various functional modules. Each state machine essentially reacts to the content of its corresponding FTQ to initiate and direct the processing activities of the state machine as shown in Figure 2. The content of FTQ includes pointers that are used to indicate where the frame data is located in the buffers. When the FTQ is empty, the state machine has no work to perform and is in the idle mode. A functional module can switch between different power-speed levels. Switching between the power-saving modes in the active state is managed by a power management policy. The DVFS controller for each functional module utilizes information about the FTQ of the module, i.e., how full the queue is and how quickly the number of entries in queue changes, to dynamically vary the supply voltage and frequency setting.

The FTQ abstraction enables high-levels of parallelism by permitting different frames in the same stage of processing to proceed concurrently. In general, the frame data is provisionally stored in memory buffers before being sent to local interconnect or network, while the control data is processed by a string of functional modules, each requiring low-latency as shown in Figure 1 (see steps (c), (d), (e), and (g)). Thus, this architecture targets the control dominated tasks rather than the storage and forwarding of the frame data. The event-queue mechanism of the FTQ enables multiple clocks and multiple voltages for the functional modules, satisfying the low-latency control data access and the high-bandwidth frame data access. The FTQ configuration for the packet receive path is illustrated in Figure 3. Functional modules i.e., QP (Queue Placement), DI (Data Initiator), and DC (Data Completion) interact with the RISC core or the memory arbiter, while transferring memory buffer pointers to the ensuing FTQ so as to advance the sequence of tasks.

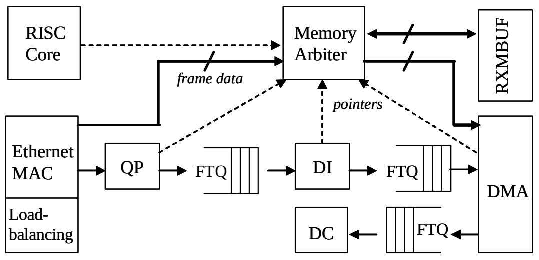


Figure 3. The configuration with FTQ for packet receive path.

The timing diagram of FTQ-based processing for the packet receive path is depicted in Figure 4 with some of FTQ-related signals. The timing diagram shows that the pointer values (e.g., C0002, C0604, C0A05, and C0C06) are transferred to the following functional modules via FTQs while performing packet header processing at each step, whereas the frame data is directly transferred to PCI-E bus from memory buffer through the DMA.

IV. Modeling a FTQ-Based System

In this section, we present a systematic approach for modeling a FTQ-based system with stochastic processes, i.e., semi-Markov decision processes (SMDP) [12]. Note that a SMDP is a tuple <*S*, *E*, *Y*, *Z*, *R*>, where *S* is a set of states, *E* is a set of actions, *Y* is the transition probability function, *Z* specifies the probability distribution of transition times for each state-action pairs, and *R* is the expected reward function [12]. Figure 5 shows the SMDP model of the Gigabit Ethernet controller for the packet receive path with a state set $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$, where *m* is the number of processing modes available to the system. This figure shows that each state in the SMDP model interacts with relevant functional modules, implying dependency between these modules. For example, the S_5 state involves RISC, QP, MA (Memory Arbiter), and RXMBUF modules. Definitions of the states for this SMDP model are provided in Table 1. The idle and sleep modes shown in Figure 5 are for the whole system, i.e., all functional modules go to sleep in S_{11}. Note also that each functional module has its own idle and sleep modes as shown in Figure 2.

The FTQ may be represented by the G/M/1 queuing model, where inter-arrival times are arbitrarily distributed and service times are exponentially distributed [13]. A general distribution is assumed for the inter-arrival times because an exponential distribution would underestimate the occurrence probability for long request inter-arrival times and so it does not adequately model the request arrival time in the idle periods [14]. The service time behavior is captured by a given service time distribution for the functional module when it is in the active mode. Similarly, the input request behavior is modeled by a given inter-arrival time distribution. Let S_i represent the ith state in a SMDP, and I_i denote the task (i.e., the job descriptor) inter-arrival time whose distribution depends only on the present state S_i. Assuming that inter-arrival times are mutually independent, we may define the arrival process of tasks at time *t* from state *i* to state *j* of the SMDP as follows:

$$a_{ij}(t) = Prob\{S_{i+1} = j,\ I_{i+1} \le t \mid S_i = i\} \qquad (1)$$

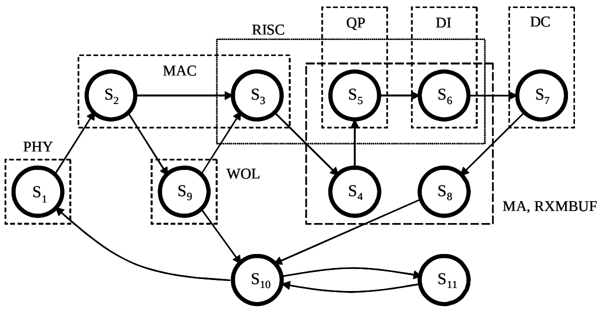


Figure 5. SMDP model of the system.

Table 1. Legend for the SMDP model of Figure 5.

State	Description
S_1	Receive data stream from physical layer interface
S_2	Perform address checking, CRC calculation, and CSMA/CD function
S_3	Calculate checksum and parse TCP/IP header
S_4	Place packet data and header into buffer memory
S_5	Buffer descriptor processing (Queue replacement)
S_6	Buffer descriptor processing (Data Initiator)
S_7	Complete buffer descriptor for packet
S_8	DMA transfers packet data to host memory
S_9	Filter WOL (Wakeup on LAN) packets during power down mode
S_{10}	System idle mode
S_{11}	System sleep mode

Let *W* denote the number of waiting tasks in the FTQ just before a new task arrives. Assuming an infinite queue size, we have:

$$q_n \triangleq Prob\{W = n\} = (1-\gamma)\gamma^n \ ,\ n = 0,1,\ldots,\infty \qquad (2)$$

where γ is the unique solution of Laplace-Stieltjes transform of the inter-arrival time distribution function [16], which is

$$\alpha_{ij}(s) = \int_0^\infty e^{-st} a_{ij}(t)dt \qquad (3)$$

Here *s* is a real-valued variable. We assume that the service times in the functional modules are exponentially distributed with the mean value of μ^{-1}. Let $T_{W.k}$ ($T_{S.k}$) represent the *waiting time* (*service time*) of the tasks in the kth FTQ and its corresponding functional module. Since the *response time*, $T_{R.k}$, of the functional module is the expected time that the tasks spend in its FTQ and in the functional module itself, we have $T_{R.k} = ((1-\gamma)\mu)^{-1}$. The waiting time in the FTQ is calculated by subtracting the service time from the response time, yielding:

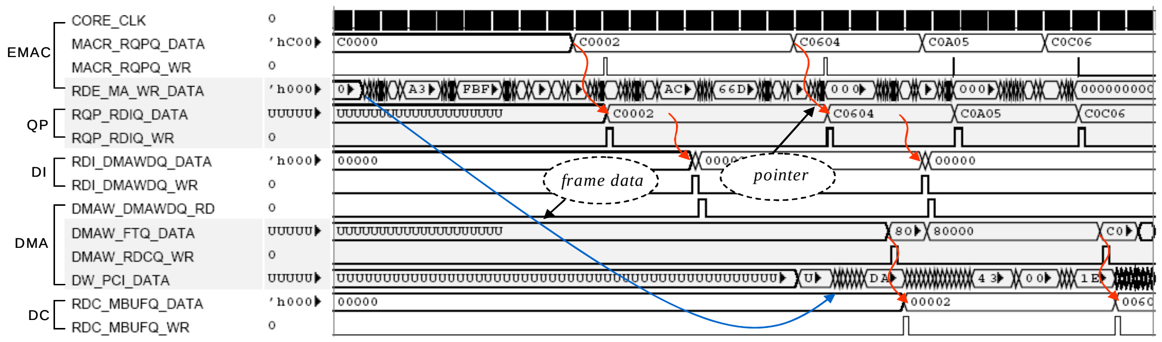


Figure 4. The simulation of FTQ-based processing for packet receive path.

$$T_{W,k} = T_{R,k} - \frac{1}{\mu} = \frac{\gamma}{\mu(1-\gamma)} \qquad (4)$$

We would like to consider the utilization of a functional module i.e., how much of the computational resource provided by the functional module is utilized by the application. More precisely, the utilization ratio, u_k, may be defined as:

$$u_k = BP \big/ (BP + IP) \qquad (5)$$

where *BP* is the duration of the busy period of the functional module, and *IP* is the duration of its idle period. Without presenting the proof, we simply state (cf. [13]):

$$BP + IP = E(T) \big/ (1-\gamma) \qquad (6)$$

where *E(T)* is the expected number of transitions in the SMDP. Thus, given the number, *n*, of tasks waiting in the FTQ, we can calculate *BP* and *IP* as follows

$$BP = \frac{E(T)}{1-\gamma} \cdot \sum_{i=1}^{n} q_i \,, \quad IP = \frac{E(T)}{1-\gamma} \cdot q_0 \qquad (7)$$

V. Performance Optimization

In this section, we present the energy optimization formulation problems and methods in multiple V_{dd}/V_{th} assignments by developing mathematical programming models.

A. Energy Optimization based on DVFS and SMDP

Let $actpow_{k.Vdd.Vth}$ (or $slpow_{k.Vdd.Vth}$) represent the power consumption of the kth functional module during its active mode running at V_{dd} and V_{th} levels (or its sleep mode). Considering the active power, we use a joint cost structure such that the expected cost rate, i.e., active power consumption, is the summation of a cost term, *k*(*s, a*), which is incurred when action *a*, i.e., the DVFS setting (*f*, V_{dd}, V_{th}), is taken, and a second term *c*(*s', a, s*), where $s, s' \in S_m$. This results in the following total cost equation: [2]

$$\begin{aligned} cost(s,a) &= k(s,a) + c(s',a,s) \\ &= pow(s) + \frac{1}{\tau(s,a)} \sum_{s \in S} Prob(s' \mid s,a) ene(s,s') \end{aligned} \qquad (8)$$

where $pow(s) = \Sigma_{k\in K}\ actpow_{k.Vdd.Vth}$, *ene*(*s, s'*) is the energy required by the system to transit from state *s* to *s'*, and $\tau(s, a)$ is the expected duration of the time that the system spent in the state *s* if action *a* is chosen, which in turn is given by

$$\tau(s,a) = \int_0^\infty t \sum_{s \in S} p_a(s,s',t) \gamma^t dt \qquad (9)$$

where γ is a discount factor, $0 \le \gamma < 1$, and $p_a(s, s', t)$ is the probability that as a consequence of choosing action *a* when the system state is *s'*, the state equals *s* after time *t*. Let a sequence of active states $s^0, s^1, \ldots, s^k$ denote a processing path δ from s^0 to s^k of length *k* with the property that $p(s^0, s^1), \ldots, p(s^{k-1}, s^k) > 0$, where *p*(*x, y*) is the probability that the system moves to state *y* from state *x* (see Figure 4). For a policy π, we define the discounted cost *C* of a processing path δ of length *k* as follows.

$$C^\pi(\delta) \triangleq \sum_{i=0}^{k} \gamma^{t_i} cost(s^i, a^i) \qquad (10)$$

[2] In this paper, subscripts denote state information whereas superscripts denote time stamp.

where t_i is the time that the system spent in state s^i before action a^i causes a transition to state s^{i+1}. Considering the expectation with respect to the policy π over the set of processing paths starting in state *s*, we may define the expected cost of the system, given that the system starts in state *s*, by $actpow^\pi_{avg}(s) = EXP[C^\pi(\delta)]$.

Ignoring the energy overhead of the transition between the active and sleep states of the system, the average energy dissipation of the functional modules can be obtained by:

$$\begin{aligned} ene_{avg} &= actpow^\pi_{avg}(s) \cdot \sum_{l\in L}\sum_{k\in K} exe_{l.k.Vdd.Vth} \\ &+ \sum_{k\in K} slpow_{k.Vdd.Vth} \cdot (T_d - \sum_{l\in L} exe_{l.k.Vdd.Vth}) \end{aligned} \qquad (11)$$

where *L* and *K* denote the set of tasks and functional modules, T_d is the given time period, and $exe_{l.k.Vdd.Vth}$ is the execution time of task *l* on functional unit *k* running at V_{dd} and V_{th}. Changing the voltage level (and correspondingly the operating frequency) of the functional modules affects the execution time of the tasks. Clearly, $exe_{l.k.Vdd.Vth} = T_{W.k.Vdd.Vth} + T_{S.k.Vdd.Vth}$. When there is a positive slack for the task to run on a functional module, DVFS can result in significant energy saving. Thus, our goal is to minimize energy consumption of a target system by choosing the optimum setting (V_{dd} and V_{th}) as a solution, subject to performance constraints:

$$\begin{aligned} \min\ & ene_{avg} \\ \text{s.t.}\ & \sum_\delta \left(\sum_{i=1}^{n} i \cdot q_i + T_{S.k.Vdd.Vth}\right) \le T_d \\ & \sum_{i=1}^{n} q_i \Big/ \sum_{i=0}^{n} q_i \ge u_k \\ & \sum_{i=0}^{n} q_i = 1 \\ & 0 \le q_i \le 1, \ \ i = 0, \ldots, n \end{aligned} \qquad (12)$$

Note that i) $\sum_{i=1}^{n} i \cdot q_i = T_{W.k.Vdd.Vth}$, ii) $T_{S.k.Vdd.Vth}$ is affected by the DVFS setting, iii) $BP/(BP+IP) = \sum_{i=1}^{n} q_i / \sum_{i=0}^{n} q_i$, and iv) u_k is a lower bound on the utilization of functional module, which is provided by the user or application.

B. Workload-Aware Multiple V_{dd} / V_{th} Assignment

Nearly all of prior work on V_{dd}/V_{th} assignment has concentrated on gate sizing [15][16] or power optimization at the circuit level [17][18]. Little attention has been paid to workload decomposition with V_{dd}/V_{th} assignment, which is the core of our approach. Initially, we perform static timing and power analysis using the standard cell libraries to determine gate delay and power values of the functional modules. To achieve accurate power values, we generate SAIF (Switching Activity Interchange File) [19] based on RTL simulation of the system. We use TSMC 130nmLP library which has 3 optional operating voltages (e.g., 1.35V, 1.5V, and 1.65V) and dual (High and Low) V_{th} for standard cells.

Our proposed multiple V_{dd}/V_{th} assignment method takes as input a circuit that has been optimized for a maximum speed by using the available slack, which is obtained by Synopsys Design Compiler. After determining the timing critical paths of the circuit, we use high supply voltage, $V_{dd.h}$, and low threshold voltage, $V_{th.l}$, for the gates on those paths. We use a low supply voltage, $V_{dd.l}$, for the other gates, especially those that drive large capacitance since this approach yields the largest dynamic power savings. Figure 6 shows the power characteristics of EthernetMAC module for various V_{dd}/V_{th} assignments (i.e., before the two assignments are combined). After redistributing the critical paths by combining high and low V_{th} cells (e.g., 1.6% usage of high V_{th} cells) for EthernetMAC, we achieve a leakage power consumption of around 7.4uW at 1.35V with 13.65ns delay. Note that cell-based design with all high-V_{th} cells consumes 5.8uW of leakage power, but gives rise to 16.2ns delay. On the other hand, an all low-V_{th} cell-based design produces 38uW leakage power with 9.36ns circuit delay. In addition to reduction in leakage power, this approach also reduces the peak power dissipation. (The peak power dissipation in a localized space can cause local heating and peak temperature). Figure 7 shows the power distribution change inside EthernetMAC module before and after multiple V_{dd}/V_{th} assignment.

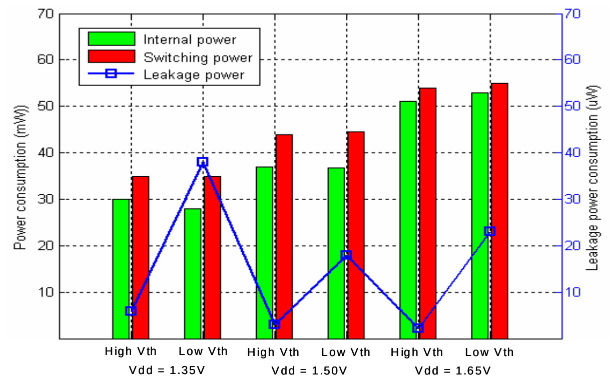
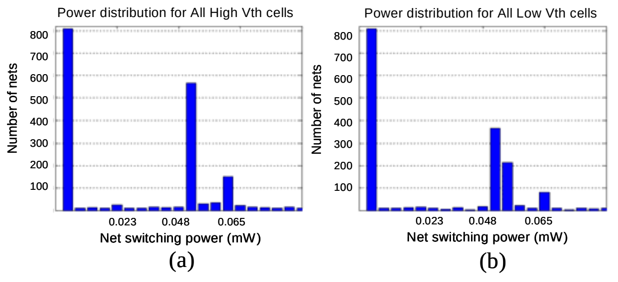


Figure 6. Power characteristics of EthernetMAC.



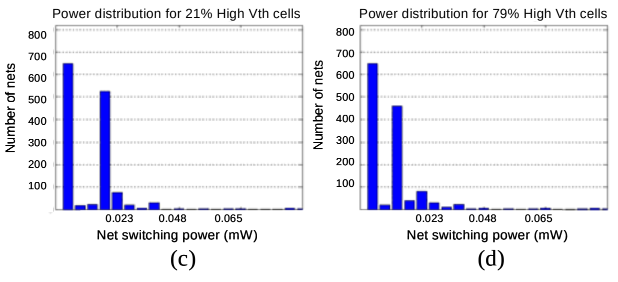


Figure 7. Power distribution inside EthernetMAC @ 1.35V, 125°C: (a) All high V_{th} cells, (b) All low V_{th} cells, (c) 21% high V_{th} cells, and (d) 79% high V_{th} cells.

Since leakage power is significant in both sleep and idle modes, we capture the energy consumed due to leakage currents as a performance metric in terms of the task workload and the utilization of functional module. Task workload, which can be represented by a queuing model as explained in section 4, is decomposed into the waiting time and the service time, where service rate μ is the function of the clock frequency and the V_{dd}/V_{th} assignment. The FTQ-based system, where queues give clues about the speed balance between the sender domain and the receiver domain, is capable of adapting the execution speed to the changing demand. Note that when combining multiple supply voltages, level converters are required each time a gate that is powered by a lower voltage level drives a gate powered by a higher voltage level. In such cases, the power and delay penalties need to be considered. Assuming no additional overhead [18], the impact of workload in terms of the consumed energy and the utilization can be characterized for applying V_{dd}/V_{th} assignment as will be seen in the next section.

VI. Experimental Results

In the first experiment, we demonstrate the performance of our designed Gigabit Ethernet controller by obtaining the throughput for streams of various packet sizes as shown in Figure 8. The figure shows that the maximum full duplex bandwidth (i.e., 1000Base–T and 100Base-T) for each packet size is achieved. The SmartBits 2000 (performance analysis system) from Spirent [20] is used to generate various packet streams, where we fixed the IP packet size for each simulation step with the inter-packet gap = 0.096us.

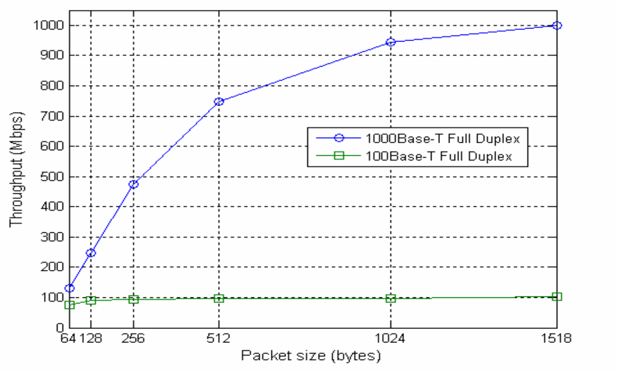


Figure 8. Throughput vs. Packet size.

The second experiment has been designed to evaluate the proposed leakage power optimization method. We analyze the leakage power dissipation after re-distributing the critical paths by using TSMC 130nmLP technology. Simulation result for EthernetMAC module in Figure 9 indicates that the performance metrics (delay and leakage power) are adjusted gradually and trade-off becomes more dramatic at the corner cases (all high-V_{th} and all low-V_{th} cell assignments).

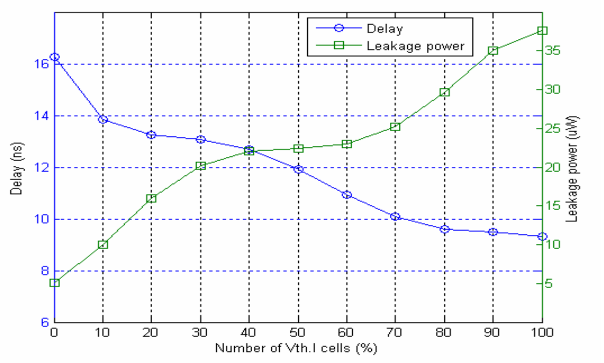


Figure 9. Trade-off: Delay vs. Leakage power (V_{dd} = 1.35V).

The third experiment is to demonstrate the effectiveness of our proposed V_{dd}/V_{th} assignment for energy optimization. Assuming that that there is no waiting time and the mean service time is μ^{-1} = 1 to simplify the experimental setup, we calculate the consumed energy by leakage power and utilization of functional module (e.g., EMAC) for various V_{dd}/V_{th} assignments and arrival rates of task as shown in Figure 10. The result demonstrates that if we assign $1.35V_{dd}/V_{th.l}$ set as an example, the energy consumed due to leakage currents in a infrequently utilized state is much greater than that in a highly utilized state, which means that $1.65V_{dd}/V_{th.h}$ assignment is the optimal solution in this case. Thus, we can see that our proposed assignment method can dynamically adjust the V_{dd}/V_{th} assignment when the workload characteristics change, which results in further energy savings.

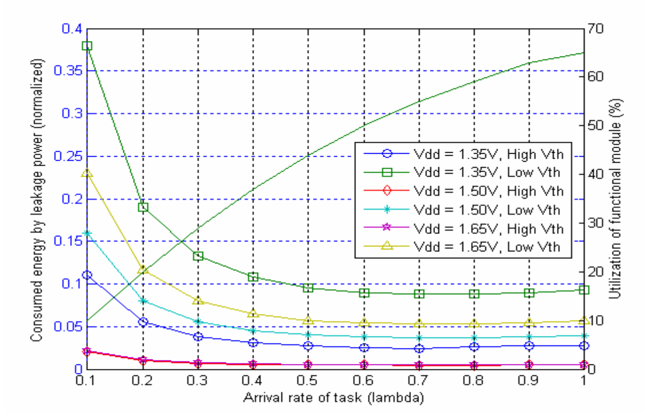


Figure 10. Energy due to leakage currents vs. Workload.

In the fourth experiment, we set the performance constraints on the T_d and u_k (e.g., T_d = 5 and u_k = 0.6) as in equation (12). The solution of the SMDP-based optimization problem produces an optimal policy. Different arrival rates (λ) of tasks are used to generate the multiple rows in Table 2, which represents the energy consumption for various V_{dd}/V_{th} assignments in the active and idle modes of the functional modules (e.g., QP, DI, and DMA). We assume that the service time is 1 to simplify the calculations. Next, we apply different workloads for each module to simulate the optimal policy as shown in Table 3. Results demonstrate that SMDP-based optimization produces energy savings for both active and idle modes up to 20% and 56%, respectively.

Table 2. Energy dissipation for various workloads (normalized).

Arrival rate	Vth	Vdd	QP		DI		DMA	
			Ene.act	Ene.idle	Ene.act	Ene.idle	Ene.act	Ene.idle
λ = 0.8	Vth.h	1.35	28.5	8.1E-4	76.3	18E-4	118.1	36E-4
		1.50	35.6	2.7E-4	94.3	6.3E-4	145.8	8.1E-4
		1.65	42.5	1.8E-4	111.4	10E-4	176.2	7.9E-4
	Vth.l	1.35	28.4	17E-4	76.2	35E-4	118.3	13E-3
		1.50	35.6	4.5E-4	94.5	10E-4	144.9	38E-4
		1.65	42.6	6.3E-4	111.2	16E-4	176.1	55E-4
λ = 0.7	Vth.h	1.35	18.2	9.0E-4	48.7	20E-4	75.4	41E-4
		1.50	22.8	3.2E-4	60.0	7.2E-4	93.1	8.0E-4
		1.65	27.1	2.1E-4	72.1	11E-4	112.4	9.2E-4
	Vth.l	1.35	18.2	19E-4	48.7	39E-4	75.5	15E-3
		1.50	22.7	4.9E-4	60.1	13E-4	93.2	43E-4
		1.65	27.0	7.1E-4	72.3	18E-4	112.4	61E-4
λ = 0.6	Vth.h	1.35	13.4	1.1E-4	36.0	22E-4	55.7	44E-4
		1.50	16.8	3.0E-4	44.5	8.1E-4	68.9	9.1E-4
		1.65	20.1	2.2E-4	54.1	13E-4	83.1	10E-3
	Vth.l	1.35	13.4	21E-4	36.2	42E-4	55.7	15E-3
		1.50	16.7	6.0E-4	44.5	13E-4	68.8	57E-4
		1.65	20.1	8.3E-4	54.1	18E-4	82.9	70E-4

Table 3. SMDP-based energy optimization (normalized).

Workload: arrival rate (λ)			Total energy (typical)		Optimal policy		Savings	
QP	DI	DMA	active	idle	active	idle	active	idle
0.8	0.7	0.6	164.5	53E-4	132.4	24E-4	20%	55%
0.7	0.6	0.5	123.9	78E-4	100.1	34E-4	20%	56%
0.6	0.7	0.8	222.6	77E-4	180.0	36E-4	19%	53%
0.5	0.6	0.7	151.4	63E-4	122.4	39E-4	19%	54%

VII. Conclusion

An FTQ-based power management technique for the Gigabit Ethernet controller was presented, where we also considered the multi-V_{dd}/V_{th} assignment problem for energy optimization. By improving the accuracy of decision making in the power management policy, performance optimizations based on DVFS and SMDP under various performance constraints were formulated and solved accordingly. Experimental results show that the proposed methods result in significant energy savings for various workloads under tight performance constraints.

References

[1] P. Willmann, H. Kim, S. Rixner, and V.S. Pai, "An Efficient Programmable 10 Gigabit Ethernet Network Interface Card," *Proc. of 11th Symposium on HPCA*, Feb. 2005.

[2] http://www.ieee802.org/802_tutorials IEEE 802.3 Tutorial. July 2005.

[3] A. Iyer, and D. Marculescu, "Power Efficiency of Voltage Scaling in Multiple Clock, Multiple Voltage Cores," *Proc. of ICCAD*, Nov. 2002.

[4] Q. Wu, P. Juang, M. Martonosi, and D.W. Clark, "Voltage and Frequency Control with Adaptive Reaction Time in Multiple-Clock Domain Processors," *Proc. of 11th Symposium on HPCA*, Feb. 2005.

[5] D.E. Lackey, et al., "Managing Power and Performance for System-on-Chip Designs using Voltage Islands," *Proc. of ICCAD*, Nov. 2002.

[6] S. Bhunia, A. Datta, N. Banerjee, and K. Roy, "GAARP: A Power-Aware GALS Architecture for Real-Time Algorithm-Specific Tasks," *IEEE Trans. on Computers*, Vol. 54, No. 6, June 2005.

[7] I. Hyunsik, T. Inukai, H. Gomyo, T. Hiramoto, and T. Sakurai, "VTC-MOS characteristics and its optimum conditions predicted by a compact analytical model," *Proc. of ISLPED*, Aug. 2001.

[8] P. Rong and M. Pedram, "Power-aware scheduling and dynamic voltage setting for tasks running on a hard real-time system," *Proc. of ASP-DAC*, Jan. 2006.

[9] http://www.pcisig.com/specification PCI-Express Spec. document.

[10] http://www.broadcom.com NetXtreme^{TM} Gigabit Ethernet Controller.

[11] Y. Hoskote, et al., "A TCP Offload Accelerator for 10Gb/s Ethernet in 90nm CMOS," *IEEE Journal of Solid-State Circuits*, Vol. 38, No. 11, Nov. 2003.

[12] M.L. Puterman, *Markov Decision Process: Discrete Stochastic Dynamic Programming*. Wiley Publisher, New York, 1994.

[13] S.M. Ross, *Introduction to Probability Models*, Academic Press, 8th edition, Dec. 2002.

[14] T. Simunic, et al, "Event-driven power management," *IEEE Trans. on Computer-Aided Design*," Vol.20, No.7, July 2001.

[15] M. Ekpanyapong, etal., "Integrated Retiming and Simultaneous Vdd/Vth Scaling for Total Power Minimization," *Proc. of ISPD*, Apr. 2006.

[16] A. Srivastava, D. Sylvester, and D. Blaauw, "Power Minimization using Simultaneous Gate Sizing, Dual-Vdd and Dual Vth Assignment," *Proc. of DAC*, June 2004.

[17] J. Wei and C. Rowen, "Implementing Low-Power Configurable Processors- Practical Options and Tradeoffs," *Proc. of DAC*, June 2005.

[18] D.G. Chinnery and K. Keutzer, "Linear Programming for Sizing, V_{th} and V_{dd} Assignment," *Proc. of ISLPED*, Aug. 2005.

[19] http://www.synopsys.com Synopsys Power Compiler Documents.

[20] http://www.spirentcom.com SmartBits document. Rev. C.